能够看到求和公式为n*(n+1)/2

```
(uintptr_t)((&((uint8_t (*) [n])0)[1+n][0]))
```

分别来拆解

1. (uint8_t (*) [n])0)，我们知道(uint8_t (*))0)这个是将0转化为uint8_t *数据类型，并且sizeof(uint8_t) = 1，那么(uint8_t (*) [n])0)表示就是申明为一个数组指针并且每行数据相比前一行数据长度要多n*sizeof(uint8_t)，内存起始地址为0，每列数据地址长度增加一个sizeof(uint8_t).实际就是行地址间隔从原来的sizeof(uint8_t)变成n*sizeof(uint8_t)
代码验证如下

```
for(i = 0; i < 10; i++) {
    for(j = 0;j < 20 ;j++) {
        printf("%4lu",(uintptr_t)(&((uint8_t (*)[5])0)[i][j]));
    }
    printf("\n");
}
```

打印结果如下：

```
  0   1   2   3   4
  5   6   7   8   9
 10  11  12  13  14
 15  16  17  18  19
 20  21  22  23  24
```

如果是uint16_t的话，由于sizeof(uint16_t)=2，所以上面每列地址数值相差2，每行相差10，如下：

```
  0   2   4   6   8
 10  12  14  16  18
 20  22  24  26  28
 30  32  34  36  38
 40  42  44  46  48
```

对于(uintptr_t)((&((uint8_t (*) [n])0)[i])，（i=0~5，n=5）：

```
  0
  5
 10
 15
 20
```

对于(uintptr_t)((&((uint8_t (*))0)[i]),(i=0~5,n=5),最常用的：

```
0
1
2
3
4
```

所以推理出了(uint8_t (*) [n])0，将行地址间隔增大了n*sizeof(uint8_t)。

2. &((uint8_t (*) [n])0)[1+n][0]）相对0地址，偏移(1+n)*n地址长度，(1+n)*1是个数，如果[1+n][1],则(1+n)*2个数。

3. (uintptr_t)((&((uint8_t (*) [n])0)[1+n][0])) 将偏移地址转化为可以输出的实际数值，即n*(1+n)，然而n(n+1)/2，就是1+2+3+...+n之和，问题得解。

4．所以能够知道[1+n][0]就是n=5，就是第6行第1列的数据为30，除以2就是sum(5)=15的数值了。

5．(uintptr_t)((&((uint8_t (*) [n])0)[1+n][0])) >> 1就是(1+n)*n/2就是1+2+3+....+n之和。